무선인터넷 플랫폼 에뮬레이터를 위한 HTTP API 설계 및 구현

박충범*, 김연수, 연대진, 유용덕, 최 훈
충남대학교 컴퓨터공학과
e-mail : {here4you,kimys,djyoun,ydyu, hchoi}@ce.cnu.ac.kr

Design and Implementation of HTTP API on Wireless Internet Platform Emulator

Choong-Bum* Park, Youn-Soo Kim, Dae-Jin Youn, Yong-Duck Yoo, Hoon Choi
Dept. of Computer Engineering, Chungnam National University

요 약

현재 국내 무선인터넷 표준 플랫폼 WIPI 2.0 규격이 제정됨에 따라 응용프로그램 개발자들을 지원하기 위한 WIPI 에뮬레이터의 구현이 필요하다. WIPI 2.0 규격은 네트워크를 통한 플랫폼의 동적 업그레이드 기능을 필수 항목으로 규정하고 있으므로 네트워크를 지원하기 위한 HTTP API 의 기능이 매우 중요하다. 본 논문에서는 WIPI C API 의 네트워크 모듈인 HTTP API 의 설계/구현 및 검증을 통하여 HTTP API 원리 적합성을 기술하고 네트워크 API 의 발전 방안을 제시한다.

1. 서론

현재 국내 무선인터넷 시장은 서비스를 제공하는 이동통신사마다 다른 플랫폼을 사용하고 있다. 이러한 현실은 컨텐츠 제공자(CP: Contents Provider)가 새로운 컨텐츠를 개발할 때 이동통신사의 플랫폼방식 따라 각기 다르게 개발해야 하는 부담이 발생한다. 이러한 문제를 해결하기 위해서 국내 이동통신 3 사의 플랫폼 방식을 하나의 규격으로 통일하는 무선인터넷 표준 플랫폼 WIPI(Wireless Internet Platform for Interoperability)가 제정되었으며 2004 년 2 월 WIPI 버전 2.0 이 발표되었다[1][2][3].

WIPI 규격의 발표와 함께 WIPI 플랫폼 규격에 맞는 응용프로그램의 개발 효율성을 향상시키기 위해 WIPI 플랫폼 규격을 준수하며, 단말기 환경과 유사한 에뮬레이터 프로그램의 필요성이 제시되어, 본 연구팀에서는 한국전자통신연구원의 지원 하에 윈도우 기반의 WIPI 에뮬레이터 프로그램을 개발하였다. WIPI 에뮬레이터는 WIPI 플랫폼 상에서 실행되는 C 기반의 응용프로그램인 Clet 의 개발을 지원하는 그래픽, 파일, 네트워크 Application Program 등의 WIPI C API 를 구현하였다[4][5].

WIPI 플랫폼에서 네트워크를 통한 응용프로그램 다운로드 및 플랫폼의 업그레이드 기능을 요구하는 만큼 WIPI 플랫폼의 네트워크 기능을 지원하기 위한 네트워크 C API 의 개발이 필요하다.

본 논문에서는 본 연구에서 설계 및 구현한 WIPI C API 네트워크 모듈 중 HTTP(Hypertext Transfer Protocol) API 를 중심적으로 기술한다.

2 장에서는 HTTP API 의 기능에 대한 기술과 함께 HTTP API 의 설계 및 구현 과정에 대해 기술하고, 3 장에서는 공인된 툴킷을 통한 HTTP API 의 검증 과정과 테스트용 Clet 을 이용한 HTTP API 의 실행 테스트를 통하여 HTTP API 의 적합성 및 활용에 대해서 기술한

* 본 연구는 정보통신부의 선도기반기술사업의 지원으로 수행되었습니다.

다. 4 장에서는 결론 및 HTTP API 의 개발 과정에서 제기된 문제점과 개선 방안에 대하여 기술한다.

2. HTTP API 설계 및 구현

2-1. HTTP API

WIPI 플랫폼은 다양한 기능의 컨텐츠를 네트워크를 이용하여 설치 및 실행하는 기능 이외에도 네트워크를 이용한 플랫폼의 업그레이드가 가능하므로 네트워크 API 매우 중요하다[3][6].

네트워크 C API 는 소켓통신을 지원하는 socket API 와 HTTP 를 지원하는 HTTP API 로 구분된다. HTTP 는 socket 을 통하여 이루어지므로 socket API 를 먼저 구현하고, 구현한 socket API 를 이용하여 HTTP API 를 구현하였다.

Socket API 는 HAL(Handset Adaptation Layer)에서 제공하는 Win32 API 를 이용하여 구현하였다. HAL 은 UI(User Interface)와 WIPI 플랫폼간의 인터페이스 역할을 함으로써 네트워크 C API 가 에뮬레이터를 통하여 네트워크를 사용할 수 있도록 지원한다.

2-2. HTTP API 설계

HTTP API 는 지원하는 기능에 따라 [표 2]와 같이 분류하였다.

<표 1> HTTP API

설명	HTTP API
HTTP 요청 준비 API	MC_netHttpOpen()
	MC_netHttpSetRequestMethod()
	MC_netHttpGetRequestMethod()
	MC_netHttpSetRequestProperty()
	MC_netHttpGetRequestProperty()
	MC_netHttpSetProxy()
	MC_netHttpGetProxy()
HTTP 요청 API	MC_netHttpConnect
HTTP 응답 관련 API	MC_netHttpGetResponseCode()
	MC_netHttpGetResponseMessage()
	MC_netHttpGetHeaderField()
	MC_netHttpGetLength()
	MC_netHttpGetType()
	MC_netHttpGetEncoding()
HTTP 해제 API	MC_netHttpClose()

- **HTTP 요청 준비 API:** HTTP 요청 단계 전에 사용되는 API 로 HTTP 식별자를 생성하고 요청 헤더의 속성들을 설정하는데 사용한다.
- **HTTP 요청 API:** 작성된 HTTP 요청 헤더를 서버로 전송하여 응답을 요청하는데 사용한다.
- **HTTP 응답 관련 API:** 서버로부터 받은 응답 메시지의 정보를 이용하는데 사용된다.
- **HTTP 해제 API:** HTTP 요청/응답을 종료하는데 사용한다.

HTTP 요청/응답 과정에서 헤더 정보의 구성에 사용되는 구조체 'httpInfo'를 정의하였으며, 구조와 기능은 다음과 같다.

```
typedef struct _httpInfo{
    M_Int32 fd
    M_Int32 appID
    M_Int32 connect
    M_Int32 socketID
    proxyInfo proxy
    M_Byte *tempBuf[MAX_HTTP_PROPERTY]
    M_Int32 tempBufID[MAX_HTTP_PROPERTY]
    M_Byte *readBuf
    M_Int32 readBufID
}httpInfo
```

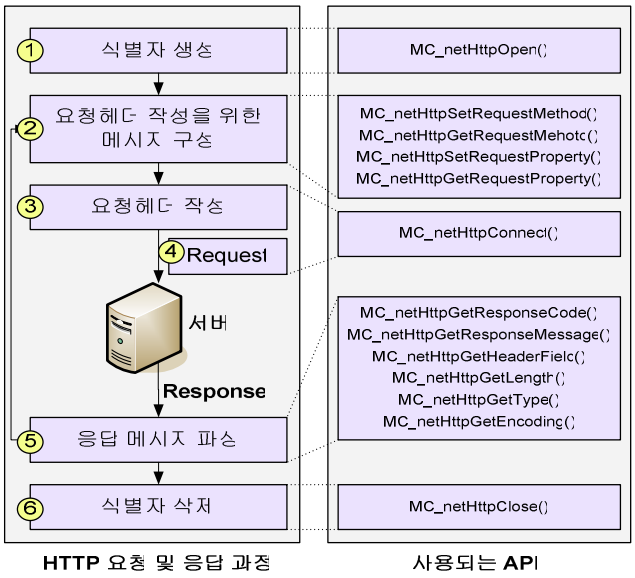
[그림 1] httpInfo 구조체

- **fd:** HTTP 식별자로 사용된다. 다중 프로그래밍에서 httpInfo 의 식별자로 사용된다.
- **appID:** httpInfo 를 사용하는 응용프로그램의 ID 를 저장하여 httpInfo 의 접근 권한으로 사용한다.
- **connect:** httpInfo 의 상태 정보를 저장한다. 상태값은 0 부터 2 까지의 정수값을 이용한다.
  - 상태값 0: HTTP 요청 준비 상태임을 의미하며, 요청 헤더 작성에 필요한 속성들을 설정하는 단계이다.
  - 상태값 1: HTTP 요청 상태임을 의미하며, 요청 헤더 속성을 제어하는 API 호출 시 에러값을 반환한다.
  - 상태값 2: HTTP 응답 상태임을 의미하며, 응답 메시지를 사용할 수 있다. 요청 헤더 속성 제어 API 및 요청 메시지 전송을 위한 API 의 호출 시 에러값이 반환된다.
- **socketID:** HTTP 요청/응답에 사용되는 소켓의 식별자를 저장한다.
- **tempBuf[]:** HTTP 요청에 사용되는 헤더의 속성 설정을 위해 사용된다. 상수 MAX_HTTP_PROPERTY 의 크기만큼의 속성을 저장할 수 있으며, index 값에 따라 설정되는 속성이 달라진다.
  - tempBuf[0]~tempBuf[2]: Method, url, HTTP 버전 정보를 설정한다.
  - tempBuf[3]~tempBuf[n-2]: 요청 헤더 속성의 설정에 사용한다.
  - tempBuf[n-1]: 요청 헤더와 함께 전송되는 데이터를 설정한다.
- **readBuf:** HTTP 응답에 사용되며 응답 메시지를 저장하는 버퍼이다.

2-3. HTTP API 구현 및 구동 원리

HTTP 는 요청 메시지(Request Message)와 응답 메시지(Response Massage)를 통하여 클라이언트와 서버 간의 Hypertext 교환을 지원하는 프로토콜이다.

HTTP API 를 이용한 HTTP 요청/응답 과정은 **[그림 3]**과 같이 설계되었다.



[그림 2] HTTP API 를 이용한 HTTP 요청/응답 과정

① 식별자 생성

MC_netHttpOpen() 함수를 이용하여 실행되며, HTTP 식별자 생성과 동시에 httpInfo 구조체에 정의한 tempBuf[]에 HTTP 요청 헤더 중 요청 라인에 해당하는 속성을 tempBuf[0]~tempBuf[2]에 저장한다.

② 요청 헤더 작성을 위한 메시지 구성

HTTP 요청 준비 API 를 통하여 HTTP 응답 메시지의 속성을 tempBuf[]에 설정하는 단계이다. 식별자 생성 단계에서 설정된 상태 라인 속성을 재설정하거나 메시지 헤더 및 메시지 바디에 해당하는 속성을 설정한다.

③ 요청 헤더 작성

HTTP 요청 API 를 통하여 요청 헤더 작성을 위한 메시지 구성 단계에서 설정한 tempBuf[]의 내용들을 조합하여 하나의 완성된 요청 헤더를 작성한다.

④ Request

MC_netHttpConnect() API 를 이용하여 요청 헤더 작성 단계에서 작성한 요청 헤더를 서버로 전송하여 응답을 요청한다.

⑤ 응답 메시지 파싱

HTTP 응답 과정으로 readBuf 를 사용하여 HTTP 응답 메시지를 저장하고, 파싱한다. 응답 메시지의 생명이 다하면 다시 요청 헤더 작성을 위한 메시지 구성 단계로 이동하거나 식별자 삭제 단계로 이동하게 된다.

⑥ 식별자 삭제

HTTP 요청/응답 서비스의 종료 단계로 식별자를 통하여 해당 HTTP 구조체를 초기화시킨다. MC_netHttpClose() API 를 통하여 이루어진다.

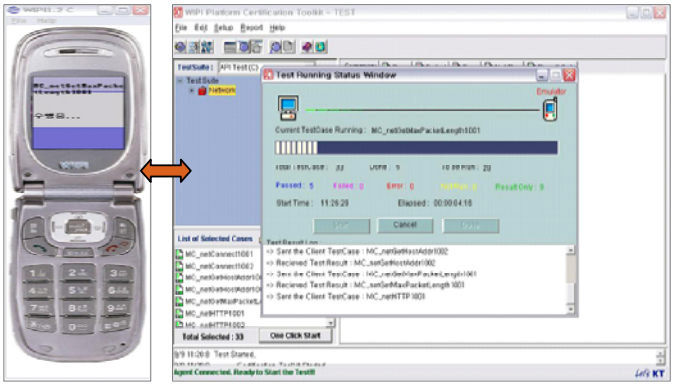
3. WIPI C API 의 검증

3-1. 네트워크 C API 의 검증

본 연구팀이 구현한 HTTP API 의 적합성은 EXEmobile㈜에서 개발한 WIPI 플랫폼 검증 툴킷인 PCT(Platform Certification Toolkit)를 이용하여 검증하였다.

PCT 를 이용한 HTTP API 의 검증 과정은 각각의 API 를 여러 경우의 파라미터 값으로 호출하게 되고, 대입한 파라미터 값에 적합한 반환값의 반환여부를 검사하는 절차를 이용한다.

[그림 4]는 에뮬레이터와 PCT 간의 통신을 통하여 테스트가 진행되는 과정을 나타낸다.



[그림 3] PCT 를 통한 검증

PCT 를 통한 HTTP API 검증은 33 가지의 항목을 통하여 이루어진다. 그 중 HTTP 요청/응답에 해당하는 주요 항목은 다음과 같다.

- **MC_netHTTP1001:** HTTP 응답 과정에서 HTTP 응답 관련 API 가 올바르게 동작하는지 검사한다.
- **MC_netHTTP1002:** HTTP 응답 과정 없이 HTTP 응답 관련 API 호출하여 에러값의 반환여부를 검사한다.
- **MC_netHTTP1003:** HTTP 요청 준비 단계에서 HTTP 요청 준비 관련 API 의 올바른 동작여부를 검사한다.
- **MC_netHTTP1004:** HTTP 요청 단계 후에 HTTP 요청 준비 API 를 호출하여 에러값의 반환여부를 검사한다.
- **MC_netHttpConnect1001:** HTTP 요청을 시도하여 성공한 경우, 성공 값의 반환여부를 검사한다.
- **MC_netHttpConnect1002:** 잘못된 식별자를 통하여 HTTP 요청을 시도했을 때, 에러값의 반환여부를 검사한다.
- **MC_netHttpOpen1001:** 인터넷 접근이 가능한 상황에서 HTTP 식별자 생성을 시도하고, 식별자의 반환여부를 검사한다.
- **MC_netHttpOpen1002:** 인터넷 접근이 가능하지 않은 상황에서 HTTP 식별자 생성을 시도하고, 에러값의 반환여부를 검사한다.

본 검증 과정에서 구현한 HTTP API 는 모든 검증 항목을 통과하였다. 이는 HTTP API 가 적합하게 구현되었음을 증명한다.

3-2. 테스트용 Clet 을 통한 HTTP API 검증

PCT 를 이용한 검증을 통해 적합성이 검증된 HTTP API 를 이용하여 실제 활용에서의 검증을 위해 한국전자통신연구원에서 제공한 HTTP API 테스트용 Clet 을 이용하여 API 의 동작을 테스트를 하였다.

테스트용 Clet 은 HTTP API 를 통하여 웹(Web)에 등록되어 있는 이미지 파일을 다운로드하고 에뮬레이터 화면에 표시하는 기능을 가지고 있으며, 본 연구에서 구현한 HTTP API 를 활용하여 실행되도록 설정하였다. **[그림 5]**는 검증 결과를 나타내고 있다.

이미지 다운로드 시도　　이미지 보기

[그림 4] 테스트용 Clet 을 통한 HTTP API 검증

다운로드 메뉴를 선택하면 테스트 서버에 등록되어 있는 이미지를 다운로드할 수 있다. 다음으로 보기 메뉴를 선택하면 다운로드한 이미지가 에뮬레이터 화면에 디스플레이 되는 것을 확인할 수 있다.

테스트용 Clet 을 통하여 본 연구에서 구현한 HTTP API 가 올바르게 실행되고 있음을 확인 수 있다. HTTP API 는 socket API 를 이용하도록 구현되었으므로 socket API 역시 원활하게 동작하고 있음을 확인할 수 있다.

4. 결론

본 연구팀은 WIPI 플랫폼의 네트워크 기능을 지원하기 위해 HTTP API 를 설계 및 구현하였다.

그러나 현재 구현된 HTTP API 는 윈도우용 WIPI 에뮬레이터를 위한 것으로서, 그 일부가 에뮬레이터의 실행환경인 윈도우에서 제공하는 라이브러리를 이용하여 개발됨에 따라 WIPI 에뮬레이터가 윈도우에 종속적이게 되는 제약이 발생하게 된다. 또한 WIPI 에뮬레이터가 컴퓨터상에서 실행되는 환경상의 문제로 인하여 WIPI 단말기가 무선네트워크를 사용하는 것과 달리 개발자의 컴퓨터 네트워크를 이용하게 된다. 하지만 WIPI 에뮬레이터를 실행하는 컴퓨터의 성능이 실제 WIPI 플랫폼 탑재 단말기의 성능보다 뛰어나고 WIPI 플랫폼 탑재 단말기에서의 무선네트워크 속도보다 WIPI 에뮬레이터에서의 컴퓨터를 통한 네트워크의 속도가 빠름을 감안해볼 때 최소한의 비용과 최소한의 시간을 이용하여 응용프로그램을 개발할 수 있는 장점이 있다.

앞으로 더 많은 개발자의 요구를 충족시키기 위해서 윈도우에 종속적인 일부 네트워크 C API 를 운영체제에 독립적으로 실행할 수 있도록 설계 및 구현하여 윈도우 외에 운영체제를 사용하는 개발자도 에뮬레이터를 사용할 수 있도록 지원할 계획이다.

참고문헌

[1] “한국무선인터넷표준화포럼” www.kwisforum.org, 2004.9.9

[2] 모바일 표준 플랫폼 WIPI 1.2.1, KWISFS.K-05-001R3

[3] 모바일 표준 플랫폼 WIPI 2.0, KWISFS.K-05-002

[4] “윈도우용 Clet 에뮬레이터 개발”, ETRI 연구보고서, 충남대학교 2004.8

[5] “휴대정보단말기용 웹 서비스 클라이언트 및 WIPI SDK”, ETRI 연구보고서, 충남대학교 2003.12

[6] 유용덕, 김연수, 최훈, “동적 Upgrade 기능을 구비한 WIPI 개발환경 설계”, 한국통신학회 추계종합 학술발표회 논문초록집, vol. 28, p.343, 2003.12

[7] “MOBILEJAVA Developer Community”, www.mobilejava.co.kr, 2004.9.9